IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

CFG based Cryptosystem for ASCII and Binary Files

[1]Manisha Gokuldas Fal Dessai,
[1]Assistant Professor,
[1]Department of Computer Engineering,
[1]Don Bosco College of Engineering, Goa, India

[2]Amey D S Kerkar,
[2]Assistant Professor,
[2]Department of Computer Engineering,
[2]Don Bosco College of Engineering, Goa, India

Abstract: The paper introduces a new symmetric key cryptosystem for ASCII and binary files. This technique makes use of context free grammar because of its cryptographic property which states it is easy to generate and validate strings from a given grammar; however it is difficult to identify a grammar given only the strings generated by it. The proposed idea comprises of following modules: Encode, Encrypt, Decrypt and Decode. Firstly the ASCII/binary file is encoded using the Base64 encode algorithm to obtain an intermediate text. To get the cipher text file the intermediate text is then encrypted using context free grammar along with the secret key, generated using random number generation algorithm. The cipher text file at the receiver side is then decrypted using context free grammar followed by Base64 decode algorithm to obtain the original ASCII/binary file.

***Index Terms* - Grammar, Cryptosystem, Symmetric, CFG, Encryption, Decryption, ASCII, Base64**

I. Introduction

Today in data communications data security is a challenging issue that touches many areas including secure communication channel, strong data encryption technique and trusted third party to maintain the database. The speedy development in information technology, the secure transmission of private data gets a great deal of attention. The conventional methods of encryption can only maintain the data security. The unauthorized user can gain access to information for malicious purpose. Therefore, it is necessary to apply effective encryption/decryption techniques to enhance data security. Strong cryptography or cryptographically strong are general terms applied to cryptographic systems or components that are considered highly resistant to cryptanalysis.

Transmission of sensitive data over the communication channel have emphasized the need for fast and secure digital communication network to achieve the requirements for secrecy, integrity and non-reproduction of exchanged information. the method for securing and authenticating the transmission of information over secure channels is provided by Cryptography . It enables us to store sensitive information or transmit it across insecure network so that unauthorized persons cannot read it.

1.1 Base64

Base64 Encoding is used to convert binary data to ASCII string format that uses 64 printable ASCII characters. The Base64 encoding is typically used for transferring email messages because email formats do not support binary data.

The data is encoded as a sequence of base-64 digits, consisting of a character from a set of 64 characters, which (starting at the character representing zero) goes in the order:

- Capital letters from A to Z
- Lowercase letters from a to z
- Digits from 0 to 9
- The characters + and /

A base-64 digit encodes six bits of the original data. Since a byte has eight bits, three bytes of the original file (24 bits) correspond to four base-64 digits. Thus, the encoding method requires you to consider each group of three bytes as a number, and express it as four digits in the base-64 system. To fill out a group of four characters padding character = is used, if not needed to encode the end of the original data.

1.2 Context-Free Grammars

Many cryptographic algorithms use one-way functions to provide their security against adversaries, but still be useful for authorized parties. A one-way function is a function that given x, it is easy to find f(x). However, given f(x) it is hard to find x. An algorithm that uses context free grammars is proposed in this paper.

A CFG consists of the following components:

- The characters of the alphabet that appear in the strings generated by the grammar are called as set of terminal symbols.
- A set of nonterminal symbols, which are placeholders for patterns of terminal symbols that can be generated by the nonterminal symbols.
- A set of productions, which are rules for replacing (or rewriting) nonterminal symbols (on the left side of the production) in a string with other nonterminal or terminal symbols (on the right side of the production).

- A start symbol, which is a special nonterminal symbol that appears in the initial string generated by the grammar.

II. Current System

The most straight-forward attack on an encrypted message is simply to attempt to decrypt the message with every possible key. Most of these attempts will fail. But one might work. At which point you can decrypt the message. Most encryption algorithms can be defeated by using a combination of sophisticated mathematics and computing power. The results are that many encrypted messages can be deciphered without knowing the key. A cryptanalyst who is skilled can sometimes decrypt encrypted text without even knowing the encryption algorithm.

The cryptographic algorithm that can be used to protect electronic data is AES which is an iterative, symmetric-key block cipher that can use keys of 128, 192, and 256 bits, and encrypts and decrypts data in blocks of 128 bits(16 bytes). Not like public-key ciphers, which use a pair of keys, symmetric-key ciphers to encrypt and decrypt data use same key. Encrypted data returned by block ciphers have the same number of bits that the input data had. Iterative ciphers use a loop structure that repeatedly performs permutations and substitutions of the input data. AES is the successor to the older Data Encryption Standard (DES).The AES algorithm is based on permutations and substitutions. Permutations are rearrangements of data, and substitutions replace one unit of data with another. AES performs permutations and substitutions using several different techniques.

III. Proposed System

The algorithm makes cryptanalysis even more difficult because of the use of “Random Number Generator” function which further decides order of encryption rounds and keys to be used to encrypt the plain text. This eliminates the overhead of defining a fixed key by the user and makes algorithm secure also. With secret key cryptography, a single key is used for both encryption and decryption. The key selection mechanism and the encoding methodology express the efficiency of the cipher text generated.

Context free grammars present the desirable cryptographic property that it is easy to generate and validate strings from a given grammar; however it is hard to identify a grammar given only the strings generated by it. The project aims at developing a CFG-based cryptosystem that will encrypt a ASCII/binary file to protect from various security attacks.

This cryptosystem will use a symmetric algorithm that will have a secret key. The ASCII/binary file will be converted into a cipher text which will be sent to the receiver who will decrypt it.

```
Procedure Key Generation ()
Input:text
Output: secret key
Begin
  Enter text
  Generate secret key
End
```

Figure 1: Algorithm for key generation

```
Procedure Encryption ()
Input: ASCII/binary file
Output: cipher text
Begin
    Base64 encode
    Key stuffing in plain text file
    Reassigning ASCII
    Count the number of characters in the
    text file
    If c mod 16= =0
        Generation of matrices
        Generate reverse productions
        Generate ASCII and binary values
        Stuffing of the bits
    Else
        Stuff space characters
        Generate cipher text
End
```

Figure 2: Algorithm for Encryption

```
Procedure Decryption ()
Input: cipher text, secret key
Output: ASCII/ binary file
Begin
    Generate binary unpacking the bits
    Generate ASCII values
    Generate reverse productions
    Reverse even odd method
    Generate matrices
    Key extraction
    Key matching
    If secret key = = key in text file
        Base64 decode
        Display plain text
    Else
        Display garbage value
End
```

Figure 3: Algorithm for Decryption

IV. DESIGN

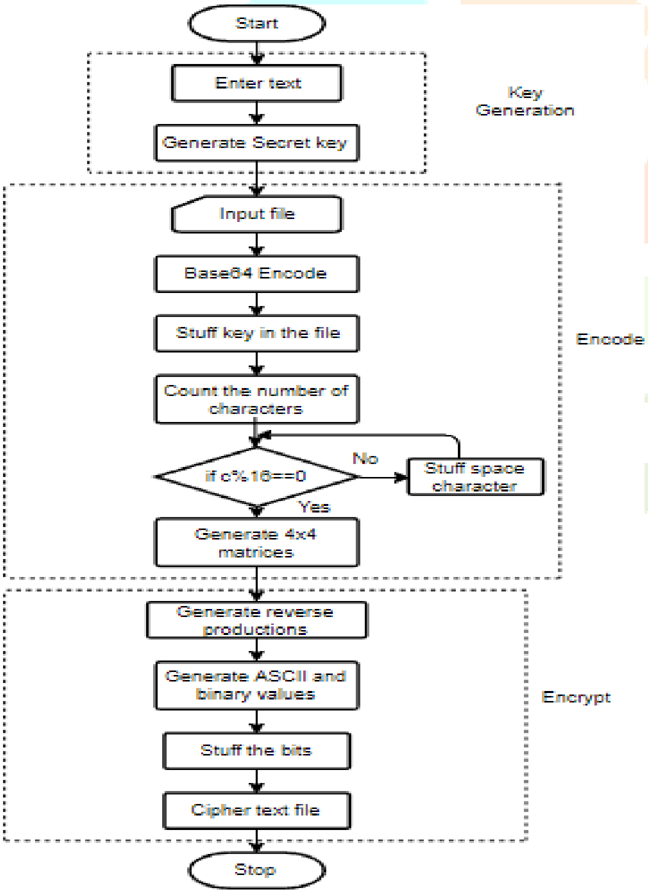


Figure 4: Flowchart for Encryption

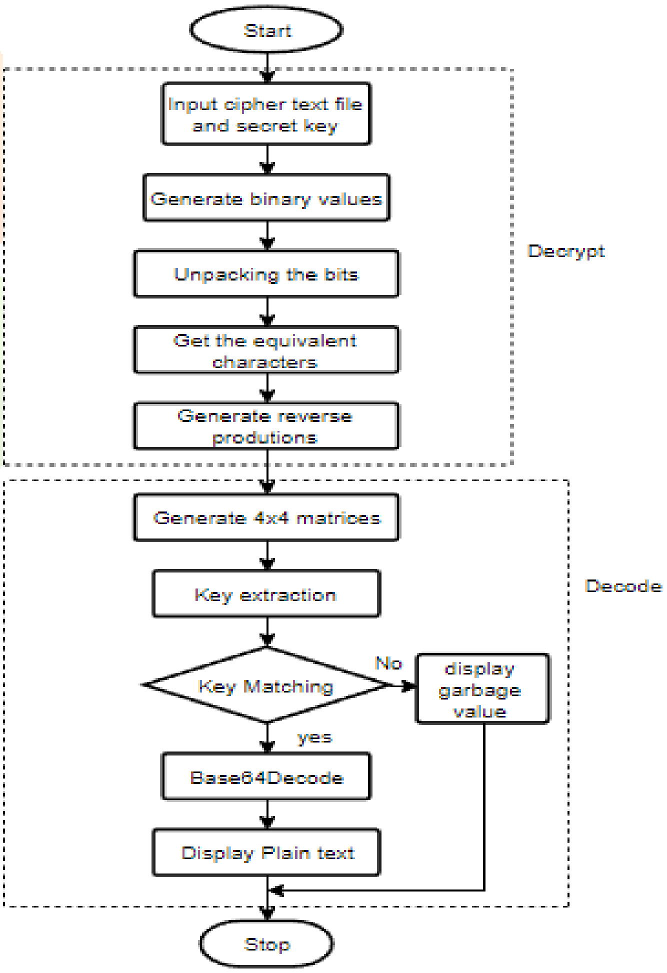


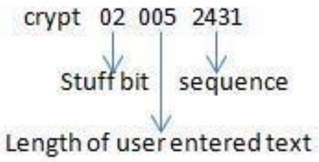
Figure 5: Flowchart of Decryption

Considering an example:

Plain text file: this is the text to be encrypted
User entered text: crypt
Generated key (secret key): crypt020052431

The first level is KEY GENERATION where the following processes will occur:

-user will enter text
-the secret key will be generated. The secret key is divided into 4 parts



02 in the secret key indicates after how many places to stuff the key,

005 is the length of the user entered text,

2431 is he matrix sequence which is randomly generated.

The next level is **ENCODE**:

- The user will input the ASCII/binary file
- Base64 Encode
- Stuffing of the key into the file
- Reassigning ASCII values
- Count the number of characters in the text file
- generate 4 X 4 matrices

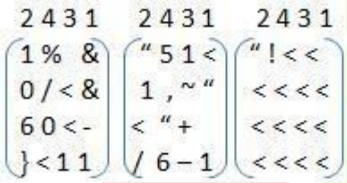
After stuffing the key we get: **VGhBcyBpcyB0aGUgdGCB0byBiZSBlbmNyeXB0ZWE**

The algorithm reassigns the ASCII values to generate the following string: 1% &0/<&60<-}<11”51<1,~”<”+ /6-1”!
It will then count the number of characters in the text file.

Let ‘c’ be the count c = 34
Now compute c mod 16
34 mod 16 = 2
16-2=14

Thus, 14 space characters will be stuffed in the text file

The next step is to generate the 4 x 4 matrices:



$$\begin{array}{ccc} \begin{array}{cccc} 2&4&3&1 \end{array} & \begin{array}{cccc} 2&4&3&1 \end{array} & \begin{array}{cccc} 2&4&3&1 \end{array} \\ \begin{pmatrix} 1 & \% & & \& \\ 0 & / & < & \& \\ 6 & 0 & < & - \\ \} & < & 1 & 1 \end{pmatrix} & \begin{pmatrix} \text{“} & 5 & 1 & < \\ 1 & , & \sim & \text{“} \\ < & & \text{“} & + \\ / & 6 & - & 1 \end{pmatrix} & \begin{pmatrix} \text{“} & ! & < & < \\ < & < & < & < \\ < & < & < & < \\ < & < & < & < \end{pmatrix} \end{array}$$

(Note: “<” is used to indicate the stuffed space characters in the above matrices just for understanding)

The next step is Shuffling of matrices hence we get the string:

1}1<&1 %-6<0&0</1/-6<”15<+””1~/<<<<<<”<!<<<<<<<<<

The third level is the ENCRYPT level which begins by generating reverse productions using context free grammar (CFG)

1}1<&1 %	-6<0&0</	1/-6<”15
A1->%1 A2 }	A4->/- A5 <6	A7->51 A8 1/

A2->11 A3 &<	A5->0< A6 &0	A8->”-A9<6
A3->E	A6->E	A9->E
<+””1~/	<<<<<<”<!	<<<<<<<<<<
A10->/ A11~<	A13->!<A14<<	A16-><<A17<<

A11->1+ A12 “”   A14->”< A15<<   A17-><< A18<<

A12->E   A15->E   A18->E

After eliminating the non-terminals we get the string:
%1}11&</-<60<&0511/”-<6/~<1+”””!<<<”<<<<<<<<<<<<

The algorithm then generates the ASCII and binary values of each character in the text file.

After bit stuffing we get:

0110001001000001111110100110001001100011010011000111100101011010011110010110110001100001011110010111101011
0000101100010011000110101111101000100101101101111000011011010100000011111101011110000110001101010111101000
1010100010101111000011110010111100001000100011110000111100001111001011110000111100101111001011110010111110
010111100001111000

This file is then converted to the cipher text:
bAúbcLyZylayMabc_D¶ðÚ•úðÆ®ŠŠðòðˆððòðòòòð

This cipher text file will then be sent to the receiver who will decrypt it using the same secret key.

The decryption process consists of the **DECRYPT** and the **DECODE** level which is exactly opposite to encryption except that key matching will occur at the end of the algorithm. If the key is matched only then the plain text will be displayed to the receiver.

V. Experimental Results

Testing for different file size:

Table 5.1: Time Comparison for Encryption and Decryption process

SIZE(KB)	TIME(ms)	
	ENCRYPTION	DECRYPTION
2	2600	2600
4	2900	2600
6	3200	2900
8	3300	2500
10	3000	2700
12	3600	3000
14	2800	2800

VI. Security attacks

6.1Brute force Attack

A brute force attack systematically attempts every possible key. It is most often used in a Known plaintext or Ciphertext-only attack. Given a finite key length and sufficient time, a brute force attack is always successful. Since we are using context free grammar it is very easy to generate productions but very difficult to get back the grammar. Secondly, we are using 128 bit key so it will require 2128 combinations which is a very big number for the attacker to try.

6.2 Cipher-text only Attack

In CFG based Cryptosystem; it is not possible to get the Original data/ Plain text if only Ciphertext is available, as there are two levels of security in it. If you try all the possible keys also, you will get the text which has no meaning.

6.3 Known Plain text Attack

The goal of a known plaintext attack is to determine the cryptographic key and possibly the algorithm which can then be used to decrypt other messages. Since we are using random key generation algorithm every time the key will be different therefore the attacker won't be able to determine the key.

6.4 Attack by breaking the Cipher text into strings

Since in CFG based Cryptosystem each production is of same length/size & since it uses random key generation it is difficult to get the plaintext.

VII. CONCLUSION

A powerful cryptosystem based on Context free grammar has been proposed in this paper. Context free grammar is used for the first time for designing a cryptosystem .The system discussed provides security without requiring additional layer of encryption like SSL and also it does not rely on any other cryptographic protocol. The salient features of the proposed algorithm include three step protocol, no large overheads, user friendliness and independent of any other cryptographic protocol. This paper presents and analyzes the protocol with respect to its robustness against malicious attacks.

The described cryptosystem makes use of interesting issues of context free grammars that until now have only been used to design programming languages. It also makes use of base64 algorithm for encoding any ASCII/binary files along with random number generation algorithm for secret key generation. Tests were then conducted to determine, given a string from a language how difficult it is to generate another string which belongs to the same language. As the size of the file increases, percentage of accepted strings generated after breaking the string decreases. Hence the chances of guessing the key and the data in the file become nearly impossible.

VIII. REFERENCES

[1] Abhishek Singh, Andre L M dos Santos, "Context Free Grammar for the Generation of a One Time Authentication Identity"

[2] Abhishek Singh, Andre L M dos Santos, "Grammar Based Off line Generation of Disposable Credit Card Numbers"

[3] Dripto Chatterjee, Joyshree Nath, Suvadeep Dasgupta, Asoke Nath, "A new Symmetric key Cryptography Algorithm using extended MSA method: DJSA symmetric key algorithm", 2011

[4] Vikrant M. Adki, Prof. Shubhanand S. Hatkar, " A Survey on Cryptography Techniques", Volume 6,Issue 6, June 2016

[5] Raj Jain, "Advanced Encryption Standard (AES)" Washington University in Saint Louis, MO 63130.

[6] Pierre L'Ecuyer1 ,"Random Number Generation" Departement d'Informatique et de Recherche Op erationnelle, Universit e deMontr eal, C.P. 6128, Canada.